IMAGE CAPTIONS GENERATION WITH CNN AND RNN

André Luis Ferreira Marques

Dept. Engª Elétrica -- Escola Politécnica – Universidade de São Paulo

ABSTRACT

This work addresses the text generation from pictures with the Flickr-30k dataset, along the use of Deep Learning networks (CNN & RNN), taken different configurations and activation functions. The BLEU score is used to evaluate the results.

***Index Terms* — Text generation, Deep Learning, RNN, CNN, BLEU, LSTM, VGG16.**

1. INTRODUCTION

The photo description using Deep Learning makes part of the Natural Language Processing (NLP), and it has gathered more importance as the computational means get stronger and specialized, with the Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) [1].

One of the applications of text generation can be seen on museums, journalism, educational textbooks, web services, among others. A recent frontier focuses the use of this digital feature to help blind people, with the application of the Braille code with Deep Learning [2].

This report deals with the text generation from selected pictures based on the joint use of CNN (training) and RNN(test), following the steps done with Python and 'Jupyter' notebook in Anaconda environment, and one refers to the code steps of [3]. The technical challenge handles the use a personal notebook computer. The code notebook is available at: https://github.com/65-1157/Aplica-o_NLP_gera-o_textos.

2. METHODS

While building up the notebook, the first task loads up the software libraries to be processed, which deal with the RNN and CNN models, Long Short-Term Memory (LSTM), Dense layers, Embedding and Dropout etc.), 'Keras', 'Tensorflow', text pre-processing, text cleaning, tokenization, text and sequence organization, lemmatization, NumPy, Pandas etc. The LSTM was used for text generation.

The VGG16 neural network architecture was used, due to its simple configuration, being also applied to image classification, object detection, general classification, image super-resolution [4], and it is already installed into 'Keras' library. Figure 1 presents the overall scheme of this neural network.

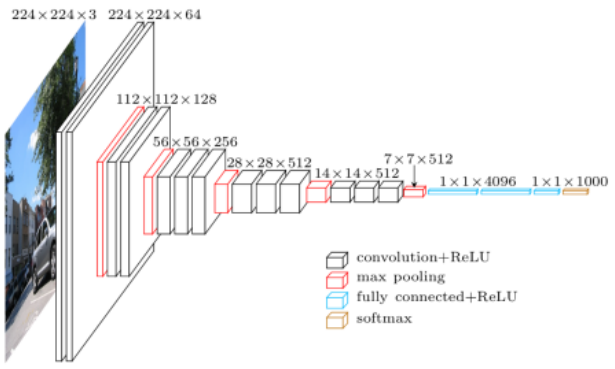

Figure 1 – VGG16 neural network general configuration

The image dataset comes from [5], which is a well-known open source with more than 30.000 images, with more than 5 years of use. For each image, five descriptions were generated in a separated file (csv, xls), which will be used to train the Deep Learning network. Figure 2 shows the two neural networks setup, dealing with two input files: images and descriptions. It is worth noting the CNN is used for the training phase and the RNN-LSTM for the test section.

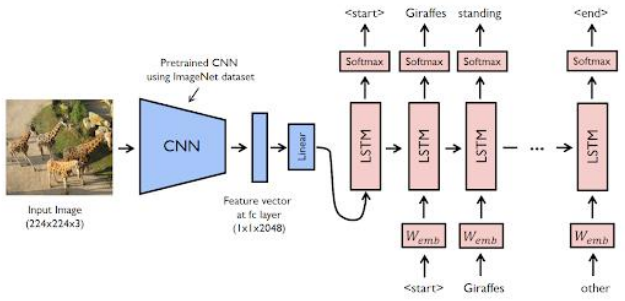

Figure 2 – Overall CNN-RNN setup [3]

The dataset is then divided into train and test files, and due to computational capacity, a personal notebook computer, one has started with a smaller number of images than the original image input file. After the descriptions loading and preprocessing, they are used to train the CNN model. The learning network has the following arrangement:

a) Input layer (224x224x3).
b) 2 Convolutional layers (224x224x64).
c) Pooling (112x112x64).
d) 2 Convolutional layers (112x112x128).
e) Pooling (56x56x128).
f) 3 Convolutional layers (56x56x256).
g) Pooling (28x28x256).
h) 3 Convolutional layers (14x14x512).
i) Pooling (7x7x512).
j) Flatten (25088).
k) 2 Dense (4096).

The maximum order of the matrices related to text descriptions, used during training, comes from the maximum length observed, which was 80 words. The activation functions selected were "Relu" and "Softmax", with optimizer "Adam" to update the weights of the network, and the 'Categorical Entropy" as the loss function, because one has used the words with the highest probabilities. In more detail, for the test phase, a RNN model is used:

a) Input (80).
b) Embedding (80x512).
c) Input (4096).
d) Dropout (80x512).
e) Dropout (4096).
f) LSTM (80x512).
g) Dense (512).
h) Dropout (80x512).
i) Dense (256).
j) LSTM (256).
k) Add (256).
l) Dense (256).
m) Dense (512).
n) Dense (570).

Five epochs were set initially to carry on the test model. The Bilingual Evaluation Understudy (BLEU) score was used to evaluate the result [6], because one has used the 'n-grams' model to produce the captions, with four sets of weights, selected by a trial-error approach, with the same neural network set:

a) 1,0,0,0.
b) 0.5,0.5,0,0.
c) 0.43,0.43,0,0.
d) 0.1,0.1,0.09,0.05.

The picture selected for captions generation follows, as shown in Figure 3, about some people playing cards, on a table covered with a green velvet sheet.

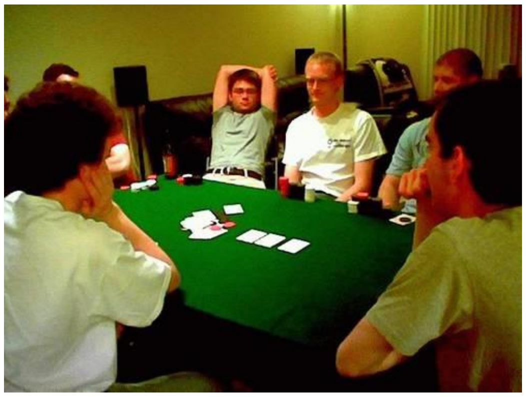

Figure 3 – Picture selected for captions generation

After the test section, the BLEU scores were:

a) Bleu_Score -1 = 0.055556
b) Bleu_Score -2 = 0.026352
c) Bleu_Score -3 = 0.043843
d) Bleu_Score -4 = 0.000000

With the same picture, the text generated was only a sequence of letter 'A', showing the need to improve the training arrangement. The loss function started at 5.56 and finished around 4.0. However, the quantity of images used was 20, taken as small on purpose due to the hardware capacity. Then, the number of epochs was increased systematically up to 120, demanding more computer work time, winding up to a loss function of 0.3 (ten times smaller than the initial value), but still with a longer letter 'A' sequence. The BLEU scores were of the same order of magnitude.

The next step focused to increase the number of pictures to 74, with 72 for train and 2 for training, for the same number of epochs. Although the same text generation outcome was obtained, along the same order of magnitude for the loss function (0.3), the BLEU scores improved about 4 or 5 times and were then:

a) Bleu_Score -1 = 0.280792
b) Bleu_Score -2 = 0.149645
c) Bleu_Score -3 = 0.168249
d) Bleu_Score -4 = 0.000000

A new set of parameters was then run, with 60 pictures for training and 14 for test, with 80 epochs. The BLEU scores were as follows, with improvement around 70%:

e) Bleu_Score -1 = 0.435583
f) Bleu_Score -2 = 0.202305

g) Bleu_Score -3 = 0.253026
h) Bleu_Score -4 = 0.000000

For each case above, the processing time has been close to 180 minutes. Nonetheless, the captions generated looked like the same outcome previously.

6. FUTURE WORK

Considering the cases developed, the suggested track deals to improve the computer capacity, with clusters like Google-Colab. A final attempt may consider the use of the notebook GPU, but it depends on the availability/capacity.

The use of more pictures and related texts improves the overall BLEU score and the quality of the text generated, say, the number of words and their connections to the selected picture. The threshold between train and test shall be close to ¼.

7. CONCLUSIONS

This work presented the steps within the caption generation for pictures, using two sorts of neural network, CNN and RNN-LSTM, in this order, to treat the digital picture first, and then to make the link with previous texts. The BLEU score was chosen to follow the performance of the build-up strategy, which included: the increase of dataset to be used, the number of epochs during the training phase of the neural networks, the number of pictures to be taken within the training and test tasks.

The BLEU scores were on average equal to 0.25, after the planned runs, with loss functions closer to 0.30, with a computing time of almost 3 hours. The captions generated were a set letters A, due to the hardware available in the end.

N. REFERENCES

[1] https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/, accessed on Sept, 1st, 2021.
[2] Zaman, Sameia & Abrar, Mohammed Abid & Hassan, Mohammad & Islam, A N M Nafiul. (2019). A Recurrent Neural Network Approach to Image Captioning in Braille for Blind-Deaf People. https://www.researchgate.net/publication/344292153.
[3] https://medium.com/swlh/automatic-image-captioning-using-deep-learning-5e899c127387, accessed on Aug, 25th, 2021.
[4] https://paperswithcode.com/method/vgg, accessed on Aug,26th, 2021.
[5] https://www.kaggle.com/hsankesara/flickr-image-dataset, accessed on Aug,30th, 2021.
[6] https://machinelearningmastery.com/calculate-bleu-score-for-text-python/, accessed on Aug. 30th, 2021.